代数的手法を用いたハードウェアの仕様記述とその詳細化について(計算アルゴリズムと計算量の基礎理論)

AUTHOR(S):

杉山, 裕二; 横山, 昌生; 北道, 淳司; 谷口, 健一

CITATION:

杉山, 裕二 ...[et al]. 代数的手法を用いたハードウェアの仕様記述とその詳細化について(計算アルゴリズムと計算量の基礎理論). 数理解析研究所講究録 1988, 666: 158-167

ISSUE DATE:

1988-07

URL:

http://hdl.handle.net/2433/100674

数理解析研究所講究録
666巻 1988年 158-167

代数的手法を用いたハードウェアの仕様記述とその詳細化について

阪大基礎工学部　杉山　裕二　(Y. Sugiyama)
横山　昌生　(M. Yokoyama)
北道　淳司　(J. Kitamichi)
谷口　健一　(K. Taniguchi)

1. まえがき

代数的記述は，その意味が合同関係のみによって簡明に定義されるので，他の手法に比べてはるかに単純・明快である．しかも，様々な抽象レベルにおいて同一の言語で記述でき，また，各レベル間の具体化の正しさの検証が形式的に行える等の利点がある．プログラムなどの仕様については既に数多くの研究成果が報告されており，ハードウェアに関しても最近，代数的手法の有効性が認識され始めている．[1,2,3]

ここでは，代数的手法を同期式順序回路を中心としたハードウェアの仕様記述に適用した場合の実現や詳細化問題について論じる．回路の記述は，抽象的順序機械[4]のスタイルで行う．まず，回路の抽象的な全状態を表すデータタイプを導入し，それを引数とする状態遷移関数及び出力関数を設定する．そして，各状態遷移後の出力関数の値を，状態遷移前の出力関数の値の複合関数として定義することにより仕様記述を行う．

ソフトウェアあるいはプログラムの仕様記述においては，“実行系”があって，項の値を計算するとき，どの順に計算するかとか，今までの結果を引数にして次にどの関数を計算するかなどは実行系が決めてくれる．しかし，ハードウェアの同期式順序回路ではクロックが供給されるごとに，次にどの状態遷移を実行すべきかは回路自身が決めなくてはいけない．状態遷移の実行順を記述するために，論理関数VALID（VALID関数と呼ぶ）を導入し，状態 s で遷移 T を実行すべきとき，VALID(T(s,…))が真，そうでないとき偽となるように書くことにする．ここでは，任意の状態sにおいて，VALID(T(S))が真となる状態遷移は高々1つ，すなわち次の状態遷移が（存在するなら）一意に決まるような順序回路を対象とする．

順序回路には，外からみて意味のある状態と，意味のある状態と次の意味のある状態に状態遷移する間の過渡的状態の2通りの状態があると考えられる．順序回路がどちらの状態であるかを表すために論理関数IS（Interested State）を導入する．

ソフトウェアの代数的記述における実現の定義は「上位レベルの記述で合同な表現式対は，下位レベルの記述でも対応する表現式対が合同になっている」というものであるが，VALID関数に関しては，そのような定義では不十分である．なぜならば，この条件だけでは，上の記述で許される状態遷移に対応する（下の記述における）状態遷移が存在するということだけを規定しており，下の記述にはそれ以外の動きが存在してもよいことになる．VALID関数は回路の動きを定めるものであるから，上の記述において順序回路が停止する（VALID関数が真にならない）時，下の記述による順序回路が停止するという条件も必要である．以下，まず順序回路と実現の定義を示し，続いてVALID関数の詳細化アルゴリズムについて述べる．

2. 順序回路

代数的言語ASL/*では[5]，従来の“項”に相当する“表現式”の集合を，文脈自由文法で指定し，表現式集合上の合同関係を公理で定義する．ASL/*には，既に定義されている合同関係に於て合同な表現式対を公理として包含する“射影”という機能がある．この機能により，記述の前提となる基本的なデータタイプや関数などの定義と，記述対象の構造や性質の定義とを分離することができる．以下では前者に関する記述を省き，後者のみについて論じる．

［定義１］（順序回路の定義）
代数的記述が次の形をしているとき，その代数的記述を順序回路と呼ぶ．

（１）前提とする基本関数（基本定数などを含む）以外の定数及び関数は次のように分類される．
(a) 初期状態を表す定数（１個のみ．INITまたはinitで表わす）
(b) 状態遷移関数T（複数．TまたはTに添字をつけたTiなどで表わす）
(c) 出力関数F（複数．FまたはFに添字をつけたFiなどで表わす）
(d) VALID関数（１個のみ．VALIDまたはvalidで表わす）
(e) IS関数（１個のみ．ISまたはisで表わす）

（２）(a)～(e)の関数に関する公理は次の形のもののみからなる．なお本稿では，簡単のため順序回路のうち状態遷移関数，出力関数の引数が状態変数のみのものだけを記述対象とする．

出力関数：
(イ) F(INIT)==C
(ロ) F(T(s))==E(s)

VALID関数：
(ハ) VALID(INIT)==TRUE
(ニ) VALID(T(s))==VALID(s) and B(s)

IS関数：
(ホ) IS(INIT)==P
(ヘ) IS(T(s))==Q(s)

但しC及びPは基本関数からなる定数式．E(s)，B(s)及びQ(S)は，状態変数s，出力関数及び基本関数のみからなる表現式

(イ)の公理は，初期状態における各出力の値を与える．(ロ)は，各状態遷移における各出力の値の変化を与える．

VALID(s)は，状態sが初期状態以降許される状態遷移のみから得られた状態であるか否かを表わしていると解釈する．よって(ハ)は，初期状態は常に許される状態であることを表わしている．VALID(T(s))という形の記述に着目するならば，これは状態sから遷移Tを行うべき条件を表していると解釈できる．(ニ)は，各状態における許される状態遷移を規定し，状態遷移の実行順を与えている．また，VALID関数の値がある状態において偽になれば，それ以降の状態ではVALID関数の値は常に偽であることを表わしている．以下，任意の状態 s において，VALID(T(s))が真となるような状態遷移（以下，VALIDな状態遷移と呼ぶ）は高々１つであるとする．

IS(s)は，状態sが着目する状態であるか否かを表していると解釈する．よって(ホ)は，初期状態が着目すべき状態かどうかを表し，(ヘ)は，状態sから状態遷移T

を行った状態が着目すべき状態かを表す条件を表していると解釈する.

順序回路の状態は，初期状態INITに何回かの状態遷移関数を施した表現式であるが，便宜上，状態を，初期状態以降その状態に至るまで適用された状態遷移関数の系列で表す. 例えば

T2(T1(INIT))

は

INIT・T1・T2

と表す. また，VALID(s)≡TRUEとなるような状態sをVALIDな状態とよぶ.

状態αに対して，初期状態からαに至るまでの状態のうちIS(α_i)が真となるような各状態α_iにおける全出力関数値の組＜F1(α_i),F2(α_i),…＞の系列をαの出力履歴と呼び，OUT(α)と書く.

3. 順序回路の実現

同じ基本関数を前提とする２つの順序回路A，Bを考える.

spec A	spec B
$F_i(INIT)==\cdots$	$f_i(init)==\cdots$
$F_i(T_j(s))==\cdots$	$f_i(t_j(s))==\cdots$
…	…
VALID(INIT)==…	valid(init)==…
$VALID(T_j(s))==\cdots$	$valid(t_j(s))==\cdots$
…	…
IS(INIT)==…	is(init)==…
$IS(T_j(s))==\cdots$	$is(t_j(s))==\cdots$
…	…

Aの各出力関数Fiに対し，$EXP_{Fi}(s)$をBの出力関数及び基本関数及び変数sのみからなる表現式とする. この時

$F_1(s)==EXP_{F1}(s)$
$F_2(s)==EXP_{F2}(s)$
…

をBからAへの出力の対応と呼ぶ. EXP=＜EXP_{F1}, EXP_{F2}, …＞とおくとき，BからAへの出力の対応を単にEXPで表すこともある. $\bar{a}$=＜a1,a2,…＞をAの出力関数値の組とし，$\bar{b}$=＜b1,b2,…＞をBの出力関数値の組とする. 各i（i=1,2,…）に対し，表現式$EXP_{Fi}(s)$中のBの出力関数f_jをそれに対応する$\bar{b}$の値b_jで置き換えて得られる式の値を$\overline{EXP_{Fi}}(\bar{b})$とする. $\bar{a}$の各i（i=1,2,…）に対し，

$a_i \equiv \overline{EXP_{Fi}}(\bar{b})$

が成り立つとき，BからAへの出力の対応EXPのもとで$\bar{a}$と$\bar{b}$が等価と言う.

これを出力履歴にも拡張する. Aの状態αの出力履歴OUT(α)と，Bの状態βの出力履歴OUT(β)が出力の対応EXPのもとで等価とは，OUT(α)とOUT(β)の組の個数が等しく，かつ履歴の順にAとBの出力関数値の組がEXPのもとで等価であることを言う.

［定義２］

２つの順序回路AおよびBと，BからAへの出力の対応EXPが与えられているものとする. そのとき，下記条件（１）および（２）が成り立つ時，かつその時のみ，EXPのもとでBはAの実現であるという.

（１）Ａの全てのVALIDな状態αに対して，Ｂの VALIDな状態βが存在し，αの出力履歴OUT(α)とβの出力履歴OUT(β)はEXPのもとで等価である．
（２）Ｂの全てのVALIDな状態βに対して，次の条件を満たすＡのVALIDな状態αとＢのVALIDな状態β'が存在する．$\beta' = \beta \cdot \gamma$と表すことができ（但し$\gamma$は，Ｂの状態遷移関数の系列で空系列を含む），$\alpha$の出力履歴OUT($\alpha$)と$\beta'$の出力履歴 OUT($\beta'$)はEXPのもとで等価である．

上記条件の（１）より，Ａの動きに対応するＢの動きが存在し，逆に（２）から，Ｂの動きに対応するＡの動きも存在する．VALIDな状態遷移が，各状態において高々１つという条件から，（２）の条件はＡが停止するときＢも停止することを意味している．

4．順序回路の詳細化の問題

ICなどは，状態遷移関数と出力関数を定義したモジュールとみることができる．そこで順序回路を構成する関数及び公理のうち状態遷移と出力関数に関するもののみからなるものをモジュールと呼ぶ．モジュールを用いた順序回路の詳細化の問題を以下のように定式化する．

> 順序回路Ａとモジュール Ｍが与えられたとき，Ｍを含む順序回路でＡの実現であるような順序回路Ｂ（出力関数を新たに導入してもよい），及びＢからＡへの出力の対応を求めることを（Ｍを用いた）ＢによるＡの詳細化という．

順序回路Ｂには，状態遷移関数と出力関数は（モジュールＭで）与えられているから，詳細化の時に追加するものは，次の３つである．
（ア）ＢからＡへの出力の対応を与える複合関数とその公理．
（イ）モジュールＭの動きを規定するVALID関数の公理とその記述．
（ウ）Ｂの着目する状態を規定するIS関数の公理とその記述．

与えられたＡ，Ｍに対して，（ア）～（ウ）を求めるために，次のような方法が考えられる．新たな出力関数が必要なら随時導入する．

（１）　Ｂ（実質はＭ）からＡへの出力の対応EXPを考案する．

（２）　Ａの各状態遷移関数T_iとＢの状態遷移との対応を表す表現式

$T1(s) == COR_{T1}(s)$
$T2(s) == COR_{T2}(s)$
…

（$COR_{Ti}(s)$は，基本関数，変数s，およびＢの状態遷移関数と出力関数からなる表現式．初期状態の対応INIT==initを含む）をＢからＡへの遷移の対応と呼び，これを考案する．全ての対応の組を$COR = \langle COR_{T1}, COR_{T2}, \cdots \rangle$とおくとき，ＢからＡへの遷移の対応を単にCORと書くこともある．遷移の対応CORの公理を用いてＡの状態αを書き換えて得られるＢの状態βを$TS_{COR}(\alpha)$（Transferred State）と書く．

（３）ＡのVALID関数の公理，出力と遷移の対応から，Ｂに対するVALID関数の公理を，必要なら新たな出力関数を導入して，順序回路の定義の（２）（ハ）及び（ニ）の形で書く．

（４）（２）で与えられた遷移の対応CORのもとで出力関数isを，次のように記述する．

順序回路ＡのVALIDな状態でIS関数が真になるような全ての状態αに対してis（TS_{COR}（α））が真となるようにし，それ以外の全てのＢの状態γに対してis(γ)が偽となるようにする．

（５）モジュールＭと（３）のVALID関数および（４）のIS関数の公理からなる代数的記述（定義よりこれも順序回路）をＢとしたとき，（１）で与えられた出力の対応のもとで，ＢがＡの実現であることを証明する．

順序回路ＡとＢの抽象度の差が大きい場合，上記（５）の証明は一般に複雑になる．そのような場合，順序回路Ａをモジュールを用いて順序回路Ｂとして実現し，さらに順序回路Ｂをモジュールより簡単な（またはより具体的な）モジュールＮを用いて順序回路Ｃとして実現するというように，段階的に詳細化していけばよい．そして，実際のIC等の機能に相当するモジュールのレベルまで詳細化が進んだ段階では，物理的な回路との対応は単純なものになるであろう．

上記（３）および（４）のVALID関数とIS関数の公理の導出は，公理の書き方に関する一定の条件のもとに自動化可能である．次節では，このうちVALID関数詳細化アルゴリズムについて述べる．なお，VALID関数詳細化の過程から，IS関数の公理も容易に導出することができる

5．VALID関数の詳細化

5・1　VALID関数の詳細化の問題

順序回路ＡとモジュールＭの記述，及び出力の対応EXPと遷移の対応CORが与えられているものとする．VALID関数の詳細化問題とはこれらから，必要なら新たな出力関数を導入して，次の２つの条件を満たすVALID関数validを求める問題である．ＡのVALID関数をVALIDで表わす．

（１）Ｂの任意の状態βに対し，valid(β)≡TRUEなら，valid(t(β))≡TRUEとなるようなＢの状態遷移関数tは高々１つである．

（２）Ａの任意の状態αに対し，VALID(α)≡TRUEなら，
valid(TS_{COR}(α))≡TRUE.

（３）Ｂの任意の状態βに対し，valid(β)≡TRUEなら，Ａの状態αが存在し，
VALID(α)≡TRUE　かつ　$TS_{COR}(\alpha)=\beta\cdot\gamma$
（但しγは，Ｂの状態遷移関数の系列で空系列を含む）

これら条件を満たすＢのVALID関数validが得られたとき，さらに，出力や遷移の対応とIS関数が“正しい”，すなわち，Ａの全てのVALIDな状態αに対し，OUT(α)とOUT(TS_{COR}(α))がEXPのもとで等価ならば，実現の定義の条件でβ，β'をTS_{COR}(α)とするように選べば，実現の条件は満たされる．従って，Ｍに関数validの記述を追加した順序回路Ｂは，EXPのもとでＡの実現になっている．

以下，VALID関数の公理の形を特定のものに限定して，VALID関数の詳細化問題に対するアルゴリズムを示す．５・２に，ここで述べるアルゴリズムの入力に関する前提を示す．

5・2　VALID関数の詳細化における前提

順序回路ＢのVALID関数validの導出において，新たな出力関数control（CONTROL関数と呼ぶ）を導入することにする．CONTROL関数は次のような意味を持つ．例えば，遷移の対応

T(s)==t2(t1(s))

を考える．これは状態遷移t1の実行が許される状態でt1を実行すれば，状態遷移t2の実行も許されることを意味する．t1の実行後，CONTROL関数の値を“状態遷移t1が実行された直後の状態”を表わすような定数に設定しておけば関数VALIDの公理を容易に導くことができる．

ここでは詳細化は段階的に，繰り返し行なうことを想定している．そこで順序回路Ａにも次のような形の公理を持つVALID関数及びCONTROL関数があるものとして，同じ形の公理を持ち，後述の条件を満たす順序回路ＢのVALID関数，CONTROL関数を導くことを考える．

アルゴリズムで扱うVALID関数とCONTROL関数の公理は次の形をしたものをいう．

（１）VALID(INIT)==TRUE

（２）VALID(T(s))==VALID(s) and {
　　(C1(s) and CONTROL(s)=STATE1) or … or
　　(Cn(s) and CONTROL(s)=STATEn)}

（３）CONTROL(INIT)==START

（４）CONTROL(T(s))==if C1(s) and CONTROL(s)=STATE1 then STATE1'
　　else if …
　　else if Cn(s) and CONTROL(s)=STATEn then STATEn'

Ci(s)は，変数s，Ｂの出力関数及び基本関数からなるBOOL式，START，STATEi，STATEi'は定数である．なお，VALID(T(s))定義式中の（VALID(s)を除く）積項“Ci(s) and CONTROL(s)=STATEi”がCONTROL(T(s))の定義式の中に現れていることに注意．

一番上のレベルの記述などで，CONTROL関数がない場合も，便宜上，CONTROL関数があるものとし，その関数の取り得る値は１つの値のみとしておく．

遷移の対応を与える表現式COR_Tについて以下の仮定をおく．tをＢの状態遷移関数，COR1，COR2を遷移の対応を与える表現式，COND(s)は変数s，基本関数及びＢの出力関数からなるBOOL式とするとき，遷移の対応の表現式COR_T(s)は次の（１）～（３）によって再帰的に構成されるもののみからなる．

（１）t(s)

（２）COR2(COR1(s))

（３）if COND(s) then COR1(s) else COR2(s)

（２）は，Ａにおいて状態遷移Tを行う時，ＢではCOR1で表される状態遷移を行った後COR2で表される状態遷移を行うことを意味し，（３）は，COND(s)≡TRUEの時はCOR1で表される状態遷移を，FALSEの時はCOR2で表される状態遷移を行うことを意味している．

遷移の対応にこのような制限を加えるのは，例えばt1，t2，t3を状態遷移関数とするとき，

if COND(t3(s)) then t1(s) else t2(s)

のように条件式に中に実行不可能な遷移(t3(s))が入るのを防ぐためである．

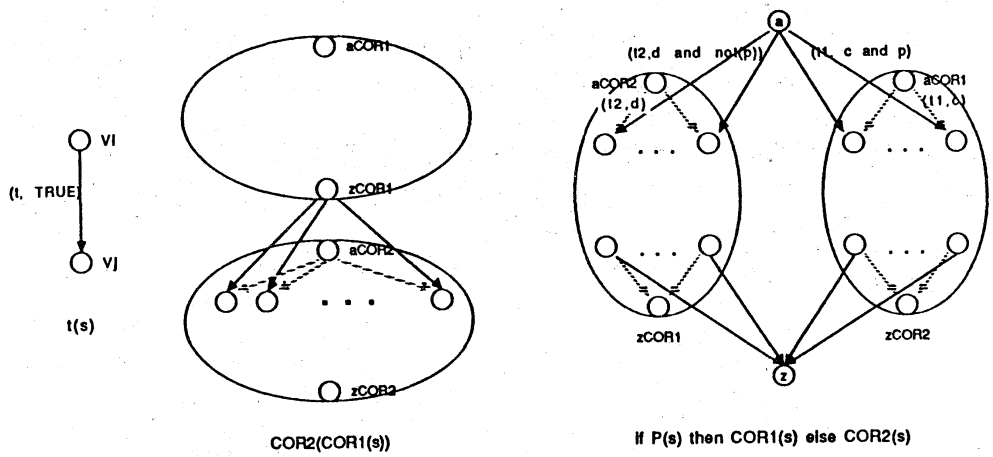
5・3　アルゴリズム

全ての $COR_T(s)$ に対して，入口節点（入射枝の存在しない節点）a と出口節点（出射枝の存在しない節点）z がそれぞれ1つであるようなラベル付き有向アサイクリックグラフ G_T =(V, A, a, z)(V は節点の集合，A は有向枝の集合. 以下 G_T を状態遷移図と呼ぶ.) を次の（1）～（3）で再帰的に定義する.

（1）t(s) の場合，v, v' を新たな節点，e を始点 v，終点 v' とする枝とすると，G_T = ({ v, v' }, { e }, v, v'). また枝 e のラベルを(t, TRUE) とする.（図1（a））

（2）COR2(COR1(s)) の場合，CORi(s)(i = 1,2)に対する状態遷移図を G_{CORi} =(V_{CORi}, A_{CORi}, a_{CORi}, z_{CORi})とし，Aを以下の手順（a）～（b）で得られる枝の集合とすると，対応する状態遷移図 G_T は，
G_T =($V_{COR1} \cup (V_{COR2} - \{ a_{COR2} \})$, A, a_{COR1}, z_{COR2}) となる.
（図1（b））
（a）初めに A = $A_{COR1} \cup A_{COR2}$ とおく.
（b）a_{COR2} から出射している全ての枝 e を A から取り除き，代わりに z_{COR1} を始点とし e と同じ終点を持つ枝 e' を追加する. e' のラベルは，e に付いていたラベルと同じものである.

（3）if COND(s) then COR1(s) else COR2(s)の場合，CORi(s)(i = 1,2)に対する状態遷移図を G_{CORi} = (V_{CORi}, A_{CORi}, a_{CORi}, z_{CORi}), a, z を新たな節点とし，A を以下の手順（a）～（d）で得られる枝の集合とすると，対応する状態遷移図は
G_T = ({ a, z } $\cup$ (V_{COR1} − { a_{COR1}, z_{COR1} })
$\cup$ (V_{COR2} − { a_{COR2}, z_{COR2} }), A, a, z) である.
(図1（c）)
（a）初めに A = $A_{COR1} \cup A_{COR2}$ とおく.
（b）a_{COR1} から出射している全ての枝 e を A から取り除き，代わりに a を始点とし e の終点である v を終点とする枝 e' を追加する. e に付いていたラベルを(t, C)とすると e' にラベル
〈t, C and COND(s)〉を付ける.
（c）a_{COR2} に対しても（b）と同じことを行なう. 但しe' にラベル
〈t, C and not(COND(s))〉を付ける.
（d）z_{COR1} または z_{COR2} に入射している全ての枝 e を A から取り除き，代わりに eと同じ始点を持ちzを終点とする枝e' を追加する. 枝のラベルはもとと同じものを付ける.



(a) (b) (c)

図 1

例えば，遷移の対応式

if P(t1(s)) then t3(t2(t1(s))) else t4(t1(s))

に対する状態遷移図は図２のようになる．

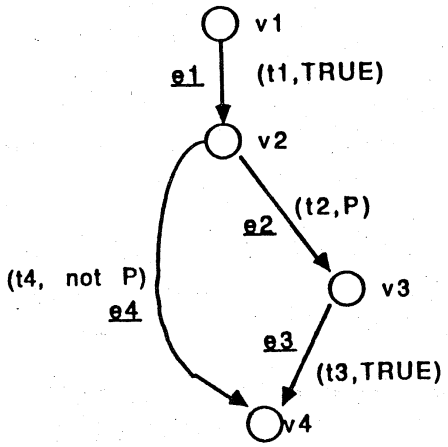


図 2

状態遷移図では，節点が状態に，枝が遷移に対応しており，枝のラベルは，遷移の名前とその遷移が実行可能となるための条件を表している．例えば，上記の例では，v1を始点v2を終点とする枝e1は，状態v1の時，遷移t1を実行し状態v2に移ることを表し，v2を始点v3を終点とする枝e2は，状態v2の時，条件Pが成り立つとき，遷移t2が実行でき状態v3に移ることを表している．

5・2に示したCONTROL関数の公理の形より，Aの各状態遷移関数Tに対して，Tが実行可能となるための条件とCONTROL関数値及びT実行後のCONTROL関数値の組の系列〈(C1(s), STATE1, STATE1'), (C2(s), STATE2, STATE2'), …〉を求めることができる.

CONTROL関数値を状態遷移図の各節点のラベルと考えると，各(Ci(s), STATEi, STATEi')はA全体の状態遷移図中の，ラベルが(T, Ci)の有向枝（始点のラベルがSTATEi, 終点のラベルがSTATEi'）を表していると考えることができる（図3(a)）. Bの状態遷移図は，Aの状態遷移図で，遷移名がTのラベルを持つ枝及びその両端の節点を（COR_Tに対応する）状態遷移図G_Tで置き換えたものである（図3(b)）.

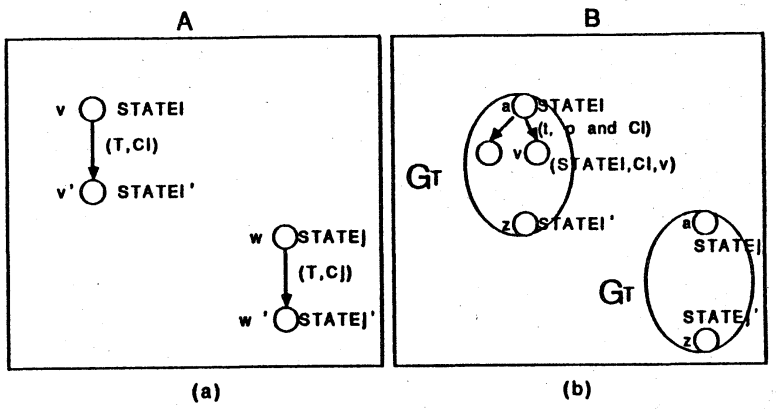


図3

各(Ci(s), STATEi, STATEi')に対する置き換えのために必要な手順は次の通りである.

（1）G_T の入口節点 a から出射する全ての枝のラベル(t, p)を (t, p and Ci(s))と置き換える.
（2）Bの CONTROL関数値に対応する各節点のラベルを次のように決める. 入口節点 a においては STATEi, 出口節点 z においてはSTATEi', a, z 以外の節点 v においては(e, v)と決める.

CONTROL関数とは，状態に対する状態名を与えるものであるから，B全体の状態遷移図において，異なる状態（節点）には異なる状態名（節点のラベル）を割り付けておけばよい. これを満足するには（2）でのBの状態名の決め方で十分である.

置き換えて得られたBの状態遷移図から，Bの各状態遷移関数tに対するVALID及びCONTROL関数の公理は次のようにすれば得られる.

状態遷移名がtのラベルを持つ各枝について，そのラベルを〈t, COND(s)〉，始点のラベル（CONTROL関数値）をST，終点のラベルをST'とすると，VALID関数には，

```
valid(t(s))==valid(s) and (
                ...
        (control(s)=ST and COND(s))or
          ...          )
```

と，CONTROL関数には，

```
control(t(s))==if ...
        else if control(s)=ST and COND(s) then ST'
                ...
```

のようにそれぞれ付け加える.

説明の都合上，（一般には無限となる）Bの状態遷移図を用いたが，実際には，AのVALID関数とCONTROL関数及び各遷移の対応COR_T（または，状態遷移図G_T）を調べるだけでよい．遷移の対応にtを含む（Aの）各状態遷移Tについて，VALID(T(s))の公理中の積項の数だけ，VALID関数とCONTROL関数の公理への上記のような追加が行われる.

6. あとがき

以上，順序回路の実現の定義と詳細化問題について述べ，VALID関数の詳細化アルゴリズムを示した．ここでは，IS関数の導出方法については省略したが，以下の考え方により容易に得られる.

順序回路AのIS関数の公理の右辺と出力の対応から，IS関数が真となるための条件は順序回路Bの出力関数で表すことができる．得られた式をeとおくと，Aの状態遷移に対応するBの一連の状態遷移が終わった直後の状態（状態遷移図においてAと同じCONTROL関数値を付けた節点に対応する状態）については，eがIS関数が真となるための条件式になっている．途中状態の場合，式eの値に関わらずIS関数を偽にしなければならない．途中状態ではCONTROL関数値を２字組にしているので，その時，IS関数の値を偽にするように，IS関数の公理を導くことはやさしい.

参考文献

[1] G.C.Gopalakrishnan, M.K.Srivas and D.R.Smith：“From Algebraic Specification to Correct VLSI Circuits”, in D.Borrione(ed.):“from HDL descriptions to guaranteed correct circuit designs”, North-Holland, pp.197-225(1987).

[2] G.C.Gopalakrishnan, D.R.Smith and M.K.Srivas：“An Algebraic Approach to the Specification and Realization of VLSI designs”, in C.J.Koomen and T.Moto-oka(eds.):“Computer Hardware Description Languages and their Applications”, North-Holland, pp.16-38(1985).

[3] S.dasgupta and J.Heinanen：“on the Axiomatic Specification of Computer Architectures”, in C.J.Koomen and T.Moto-oka(eds.):“Computer Hardware Description Languages and their Applications” North-Holland, pp.1-15(1985).

[4] 嵩，谷口，杉山：“代数的言語の設計と処理系”，榎本編：“ソフトウェア工学ハンドブック”，オーム社(1986-06).

[5] 嵩，谷口，杉山，関：“代数的言語ＡＳＬ／＊－意味定義を中心に－”，信学論（Ｄ），J69-D,7, pp.1066-1074(1986-07).

[6] 横山，谷口：“ハードウェアの代数的仕様記述とその具体化について”，信学技報，Vol.87,No.181,COMP87-34,(1987-09).